

FPGA Design Tutorial

Version 5.0 – Fall 2006

Updated Tutorial: Jake Adriaens

Original Tutorial: Matt King, Surin Kittitornkun and Charles R. Kime

Table of Contents

1. Introduction and Preparation.....	2
2. Project Creation.....	2
3. Design Entry.....	4
4. Behavioral Simulation.....	5
5. Translate and Post-Translate Simulation.....	6
6. Map, Place & Route, and Post-Place & Route Simulation.....	6
7. Generating the FPGA Programming File.....	7

Changes to V.3.0

- Post-synthesis simulation has been added.
- Flow diagrams have been added.

Changes to V.3.1

- Design Synthesis: Adding Verilog cores (*.v) to FPGA Express instead of EDIF (*.edn)
- This can solve some unexpected synthesis problems when the design is more complex and help identify the Warnings in FPGA Express.

Changes to V.4

- Updated for Xilinx Foundation 4.2i and Modelsim 5.6d

Changes to V.4.1

- Updated for Modelsim 5.7e

Changes to V.5.0

- Updated for Xilinx ISE 6 and Modelsim XE II 5.8c
- Regenerated cores with new Xilinx libraries

1. Introductions and Preparation

The FPGA design flow can be divided into the following stages:

1. Design Entry
 - a) Performing HDL coding for synthesis as the target (Xilinx ISE)
 - b) Using Cores (Xilinx Core Generator)
2. Behavioral Simulation of synthesizable HDL code (ModelSim)
3. Design Synthesis (Translation) (Xilinx ISE)
4. Design Implementation (Map, Place & Route) (Xilinx ISE)
5. Timing (Post Implementation) Simulation (ModelSim)

This design flow is based on the assumption that the student:

1. Is familiar with HDL coding using either Verilog HDL or VHDL.
2. Recognizes the difference between HDL coding for synthesis and for simulation.

The final ECE 554 project usually contains a large/complex subsystem, which takes a long time to synthesize (>10 min.) and is unchanged or infrequently changed during the system debugging loop.

Preparation

- Log on a Windows workstation (in 3628 Engineering Hall only).
- Create "I:\ece554\tutorial" directory
- Download and unzip the files from http://homepages.cae.wisc.edu/~ece554/new_website/Tutorial/tutorial_files.zip into the directory.

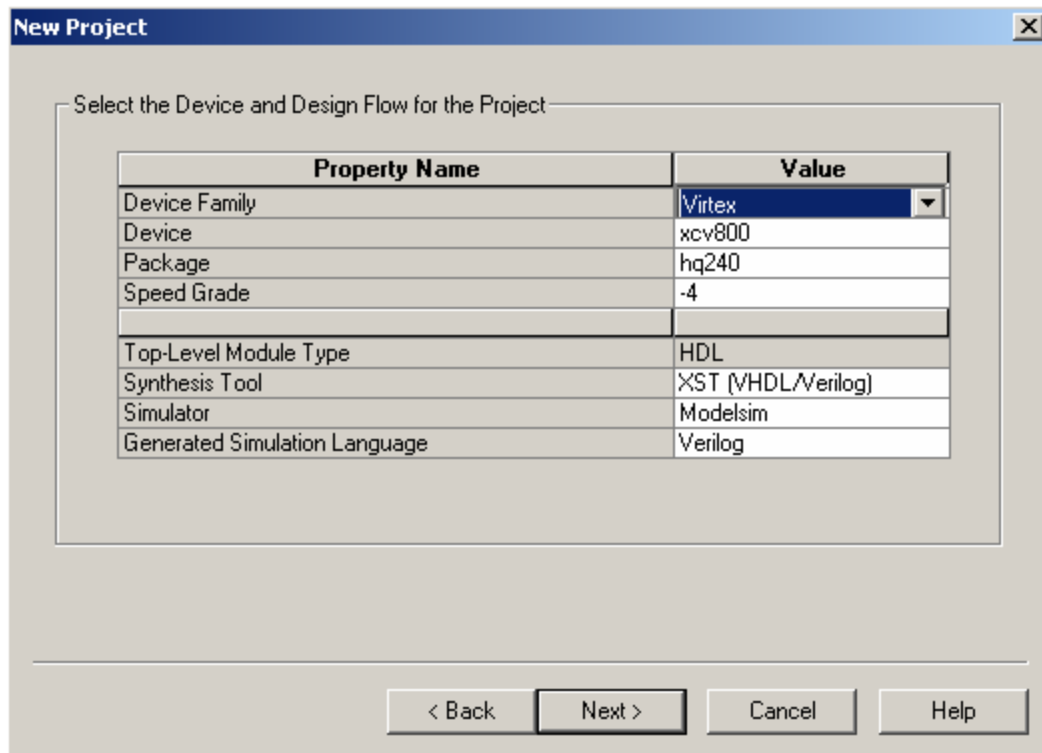
File name	Detail
mltring.v	The top most file contains the "mltring" module and other interfaces.
mac.v	The top-level file contains "mac_test" module. (MAC = Multiply-ACcumulate)
mltring.ucf	User constraint file contains port names and their corresponding pin location assignments.
mltring_tb.v	Testbench to simulate "mltring.v" in ModelSim
mult_4x4.*	4-by-4 bit multiplier core: .v for functional simulation, .edn for netlist
reg8b.*	8-bit register core: .v for functional simulation, .edn for netlist

2. Project Creation

- 2.1) Start Xilinx ISE 6 – Project Navigator (Icon on the desktop)
- 2.2) Select File->New Project
- 2.3) Name the project *tutorial*, place it in *I:\ece554\tutorial*, make sure top-level module is set to *HDL*, then click next.

Note: The Xilinx toolset does not support spaces in file or directory names.

2.4) This screen describes the hardware and simulation environment we will be using. Make sure your settings match those shown below, then click next.



2.5) This screen allows you to create new files to add to the project, we will not create any new files for the tutorial, click next.

2.6) Here is where we add existing source files, click add source.

2.7) Browse to *I:\ece554\tutorial* and select *mult_4x4.xco*, *reg8b.xco*, *mltring.v*, *mltring_tb.v*, *mltring.ucf*, *mac.v* then click open.

Note: You may select multiple files by holding the control key when clicking on them.

2.10) At this time you will be prompted to specify what type of file the verilog sources are. Set them as follows:

mac.v - Verilog Design File

mltring.v - Verilog Design File

mltring_tb.v - Verilog Test Fixture File

Note: A verilog design file is a synthesizable verilog source file which is part of your design, and cannot be simulated by itself.

A verilog test fixture file is a verilog source file whose purpose is to test another verilog module and will not be synthesized with the rest of the design.

2.11) Now uncheck copy to project for all the files, then click next.

2.12) You will be presented with a summary of the project, click finish.

2.13) Lastly, you will be asked to associate the ucf file with a source, select *mltring* and click ok.

2.14) The project has now been created.

3. Design Entry

3.1) At this time we will add a new module to the project with the core generator.

Note: The core generator is a tool that is able to automatically generate modules to include in your design. It is common to use the core generator to create register files, adders, multipliers, etc.

3.2) In the Sources in Project window on the top left, notice the ? symbol next to adder. This indicates that the module in the hierarchy above adder, in this case mac, has instantiated a module named adder but that module does not exist.

3.3) Double click on the mac module, this will open it in a text editor on the right of the screen.

3.4) Notice on line 13 the adder module is instantiated.

3.5) To create the adder module, double click on adder in the Sources in Project window.

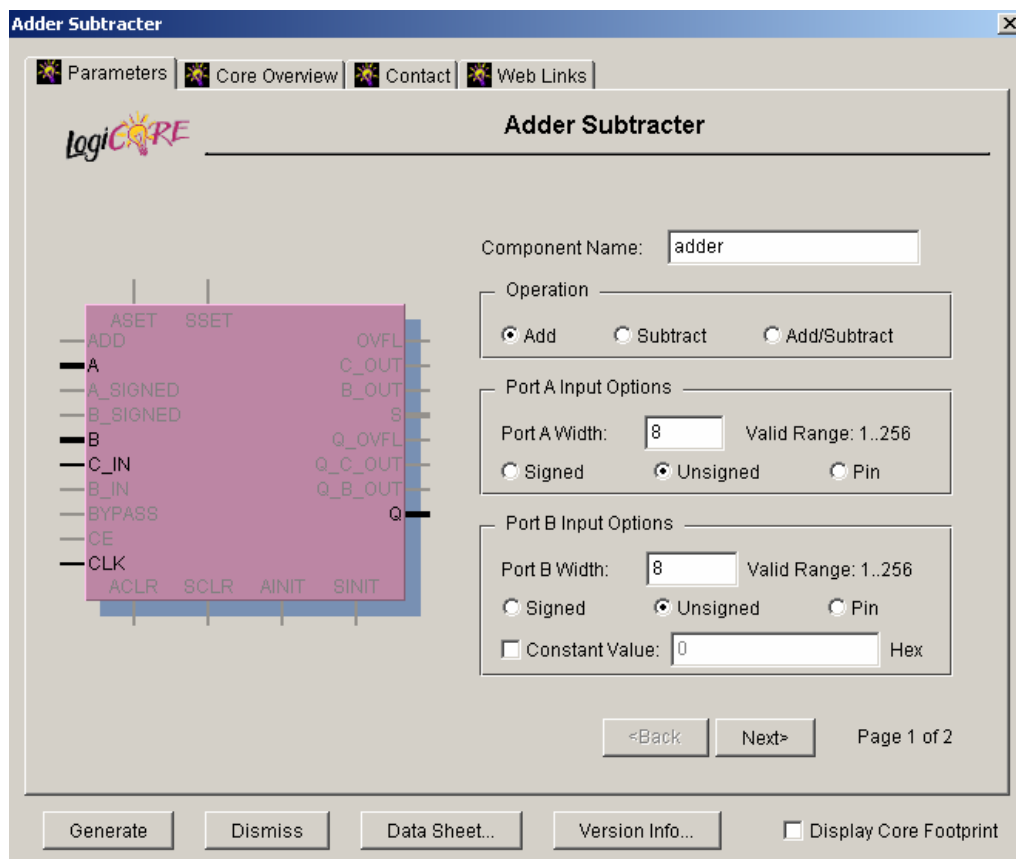
3.6) You are presented with the New Source dialog. Select *IP (CoreGen)*, name the file *adder*, set the location to *I:\ece554\tutorial*, make sure the Add to Project option is checked, then click next.

3.7) The next dialog prompts you to select the type of core we will be creating. Select *Math Functions->Adders & Subtracters->Adder Subtractor*, then click next.

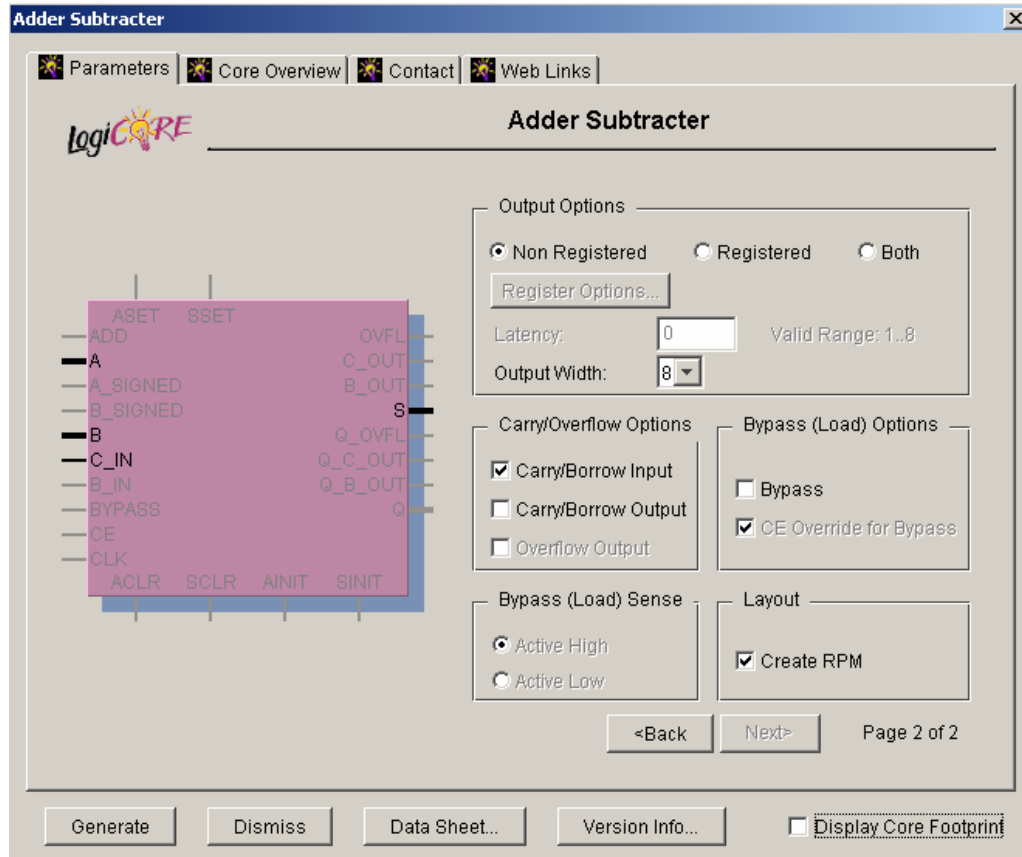
3.8) A summary of the core is presented, click finish.

3.9) The next dialog allows you to specify details of your adder subtracter. The datasheet button will open a pdf with a detailed description of the functionality of the core if you desire more information about it.

3.10) Set the options for the adder as shown below, then click next.



3.11) Set the options for the adder as shown below, then click generate to create the adder.



3.12) On successful generation of the adder click ok. The adder module should no longer have a ? symbol next to it.

4. Behavioral Simulation

4.1) Behavioral simulation is testing of your design before it has been synthesized, to test a verilog model, you require a testbench that instantiates the module and drives its inputs. In our case we will test the module mltring with the testbench mltring_tb. Select mltring_tb in the sources in project window.

4.2) In the process window below the sources in project window, we are given options related to the testbench file. Right click simulate behavioral model and select properties.

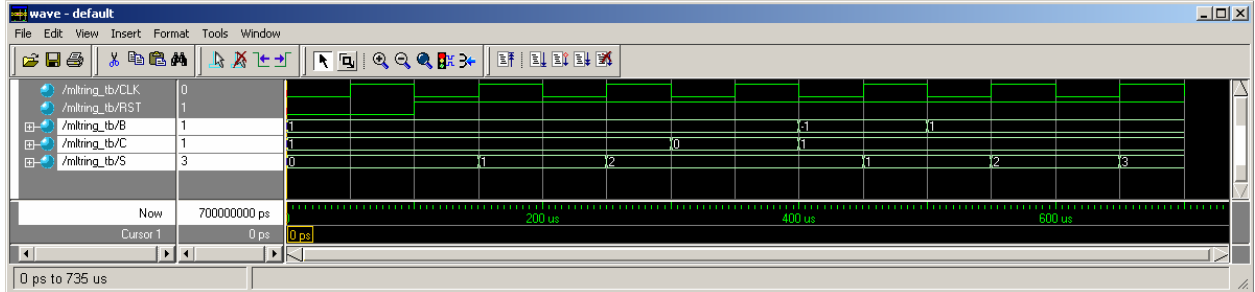
4.3) In the properties window change simulation run time to 700000ns, then click ok.

4.4) To begin simulation double click simulate behavioral model, this will start modelsim.

4.5) Maximize the modelsim wave window and zoom full (zoom full is the blue magnifying glass).

4.6) Select B, C, and S then right click and select radix->decimal.

4.7) The waveform should look similar to the figure below. Close modelsim.



5. Translate and Post-Translate Simulation

- 5.1) The translate step takes the verilog code you have written and converts it into a primitive gate representation. Poorly written verilog will often generate problems such as latches in this step. In the sources in project window select mltring.
- 5.2) In the processes for source window expand Implement Design and double click translate. This will begin the translate process.
- 5.3) When translate is complete, select the error tab at the bottom of the Xilinx project navigator window. There should be no errors.
- 5.4) Select the warnings tab at the bottom of the Xilinx project navigator window. There may be some warnings, generally warnings about unrecognized directives and unknown properties are safe to ignore.
- 5.6) To run post-translate simulation select mltring_tb in the sources in project window, and then double click simulate post-translate verilog model in the processes for source window.
- 5.7) Modelsim will start and run a simulation, verify the waveform is the same as before and then close modelsim.

6. Map, Place & Route, and Post-Place & Route Simulation

- 6.1) The mapping step takes the translated verilog and maps it to a specific technology library. The place & route step takes that mapping and determines where gates will be placed on the fpga and how they will be connected. Post-place & route simulation allows you to simulate a design with timing information such as gate delays included, which may illuminate errors in the design that previous steps did not. In the sources in project window select mltring.
- 6.2) In the process for source window under implement design, double click place & route, this will start mapping, and if mapping succeeds start place & route.
- 6.3) Upon completion, check for errors, there should be none. Then check for warnings, it is safe to ignore warnings about bel_d_min_period.
- 6.4) In the processes for source window expand map, and double click map report, this will bring up the mapping report in the window on the right. The mapping report allows you to see how much of the fpga resources your design is using, such as flip-flops, look-up-tables (LUTs), and slices (logic gates).

- 6.5) In the processes for source window expand place & route, then expand generate post-place & route static timing, then double click text based post-place and route static timing report. This will open the report in the window on the right and provide useful timing information about your design.
- 6.6) To begin post-place & route simulation select mltring_tb in the sources in project window and double click simulate post-place & route verilog model in the processes for source window.
- 6.7) Verify that the modelsim waveform is similar to before and then close modelsim.

7. Generating the FPGA Programming File

- 7.1) After the design is working correctly in post-place & route simulation, you can generate a programming file that can be used to setup the fpga to physically run your design. Select mltring in the sources in project window.
- 7.2) In the processes for source window double click generate programming file.
- 7.3) When creation of the programming file is complete it will be located at:
I:\ece554\tutorial\mltring.bit